Optimal Online Deterministic Algorithm and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers

Anton Beloglazov and Rajkumar Buyya
CLOUDS Lab, Dept. of Computer Science and Software Engineering
The University of Melbourne, Australia

Concurrency and Computation: Practice and Experience 2011

Outline

- Introduction
- System Model
- Adaptive Heuristics for Dynamic VM Consolidation
- Performance Evaluation
- Conclusion and Future Directions

Introduction (1/2)

- Although servers are usually not idle, most of the time servers operate at 10-50% of their full capacity. Even idle servers still consume about 70% of their peak power.
  - Keeping servers underutilized is highly inefficient.

- Virtualization allows Cloud to create Virtual Machine (VMs), improving the utilization of resources. However efficient resource management is not trivial.

Introduction (2/2)

- Aggressive consolidation of VMs can lead to performance degradation. If the resource requirements are not fulfilled, the application can face increased response time, time-outs or failures.

- The focus of this work is to reduce energy consumption of a virtualized data center, while meeting the Service Level Agreement (SLA).

System Model (1/4)

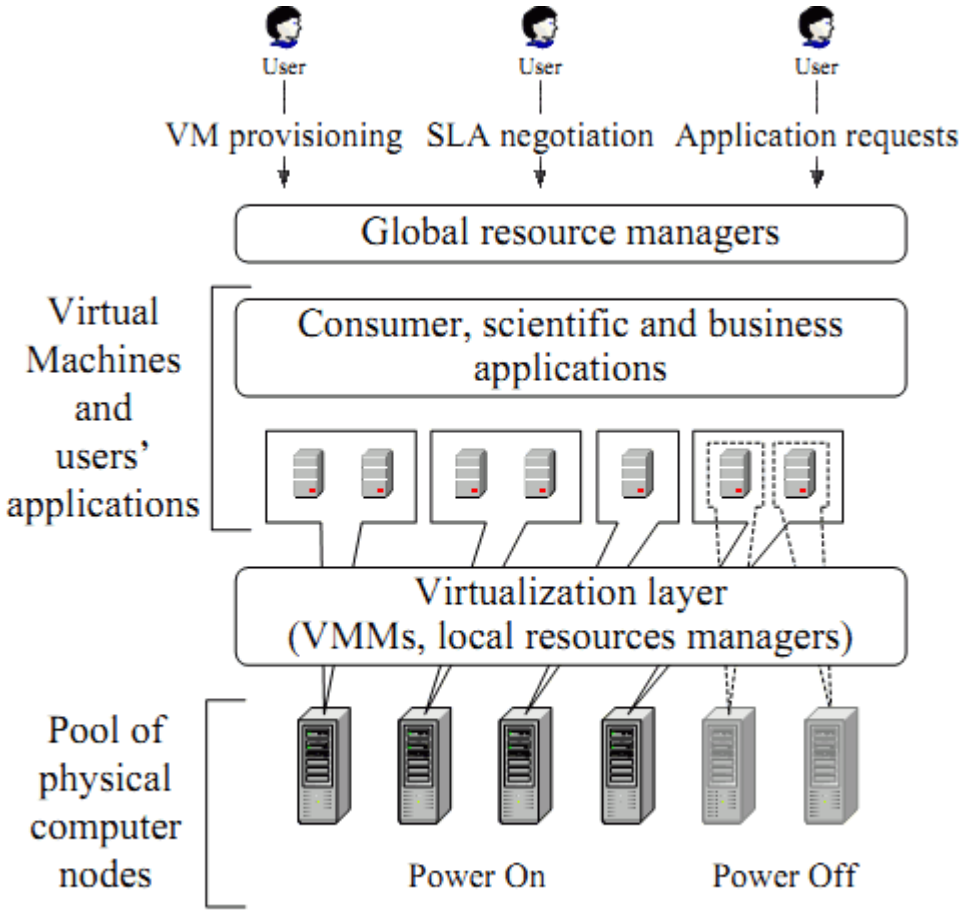


Figure 1. The system view

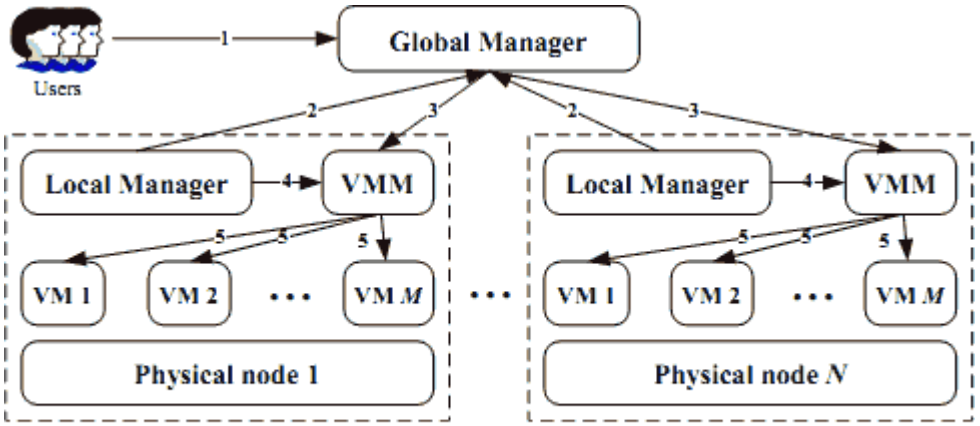


Figure 2. The system model

System Model (2/4)

- Power Model
  - The power consumption can be described by a linear relationship between the power consumption and CPU utilization even when DVFS is applied.
  - We utilize real data on power consumption provided by the results of the SPECpower benchmark.

Table I. Power consumption by the selected servers at different load levels in Watts

Server	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP ProLiant G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP ProLiant G5	93.7	97	101	105	110	116	121	125	129	133	135

System Model (3/4)

- Cost of VM Live Migration
  - The average performance degradation can be estimated as approximately 10% of CPU utilization.
  - Each VM migration may cause some SLA violation. Therefore it's crucial to minimize the number of VM migrations.

$$T_{m_j} = \frac{Memory_j}{Bandwidth_j}$$

$$U_{d_j} = 0.1 * \int_{t_0}^{t_0+T_{m_j}} u_j(t)\, dt$$

System Model (4/4)

- SLA Violation Metrics
  - SLA Violation Time per Active Host

$$SLATAH = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{s_i}}{T_{a_i}}$$

  - Performance Degradation due to Migrations

$$PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{d_j}}{C_{r_j}}$$

  - SLA Violation

$$SLAV = SLATAH \cdot PDM$$

Adaptive Heuristics for Dynamic VM Consolidation (1/13)

- Dynamic consolidation problems:
  1. Determining when a host is considered as being overloaded
  2. Determining when a host is considered as being underloaded
  3. Selection of VMs that should be migrated from an overloaded host
  4. Finding a new placement of VMs selected fro migration from overloaded and underloaded hosts

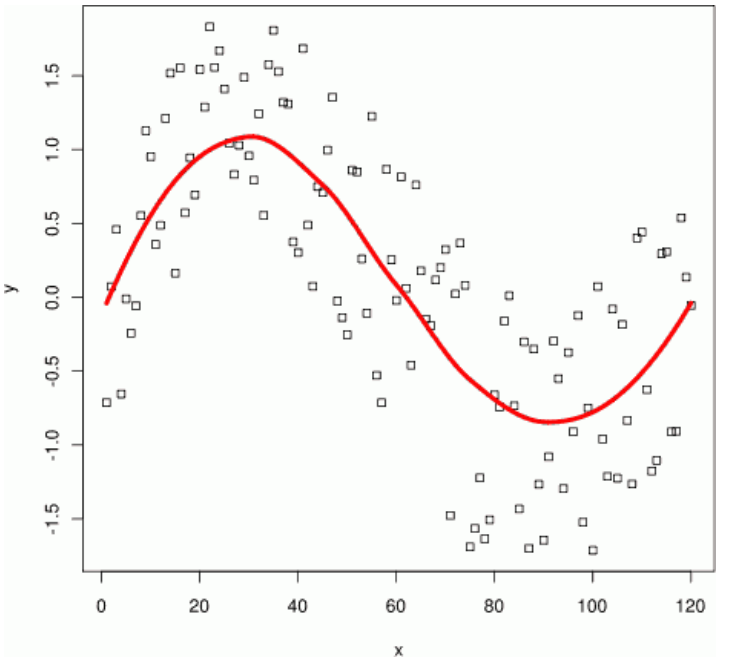
Adaptive Heuristics for Dynamic VM Consolidation (2/13)

Algorithm 1: VM placement Optimization

```
1  Input: hostList   Output: migrationMap
2  foreach host in hostList do
3  |   if isHostOverloaded (host) then
4  |   |   vmsToMigrate.add(getVmsToMigrateFromOverloadedHost (host)
5  migrationMap.add(getNewVmPlacement (vmsToMigrate))
6  vmsToMigrate.clear()
7  foreach host in hostList do
8  |   if isHostUnderloaded (host) then
9  |   |   vmsToMigrate.add(host.getVmList ()
10 |   |   migrationMap.add(getNewVmPlacement (vmsToMigrate))
11 return migrationMap
```

Adaptive Heuristics for Dynamic VM Consolidation (3/13)

- Host Overloading Detection
  - Adaptive Utilization Threshold
    - Median Absolute Deviation (MAD)
    - Interquartile Range (IQR)
  - Regression
    - Local Regression (LR)
    - Robust Local Regression (LRR)



Adaptive Heuristics for Dynamic VM Consolidation (4/13)

- Median Absolute Deviation (MAD)
  - For a historical utilization data set: $X_1, X_2, \cdots, X_n$
  - The median of the absolute deviations from the data's median

$$MAD = median_i\left(\left|X_i - median_j\left(X_j\right)\right|\right)$$

  - $s \in R^+$ is a parameter that defines how aggressively the system consolidates VMs.

$$T_u = 1 - s \cdot MAD$$

Adaptive Heuristics for Dynamic VM Consolidation (5/13)

- Interquartile Range (IQR)
  - The difference between the third and first quartiles
  - Also called middle fifty

$$IQR = Q_3 - Q_1$$

$$T_u = 1 - s \cdot IQR$$

Adaptive Heuristics for Dynamic VM Consolidation (6/13)

- Local Regression (LR)
  - The observations (x_i, y_i)
  - LOESS, proposed by Cleveland (1979)

$$T^*(u) = (1 - u^3)^3 \qquad 0 \le u \le 1$$

$$w_i(x) = T^*\left(\frac{\Delta_i(x_k)}{\Delta_1(x_k)}\right) = \left(1 - \left(\frac{x_k - x_i}{x_k - x_1}\right)^3\right)^3$$

  - We can find a trend line $\hat{g}(x) = \hat{a} + \hat{b}x$, we estimate the next observation $\hat{g}(x_{k+1})$

$$\min \sum_{i=1}^{n} w_i(x)(y_i - a - bx_i)^2$$

  - The host is considered overloaded if the inequalities are satisfied:

$$s \cdot \hat{g}(x_{k+1}) \ge 1, \qquad x_{k+1} - x_k \le t_m$$

Adaptive Heuristics for Dynamic VM Consolidation (7/13)

- Robust Local Regression (LRR)
  - Robust estimation method *bisquare*
  - $\widehat{\epsilon_i} = y_i - \widehat{y_i}$

$$Weight\text{: } w_i(x) \rightarrow r_i w_i(x)$$

$$r_i = B(\frac{\widehat{\epsilon_i}}{6s})$$

$$B(u) = \begin{cases} (1-u^2)^2 & if |u| < 1 \\ 0 & otherwise \end{cases}$$

$$s = median|\widehat{\epsilon_i}|$$

Adaptive Heuristics for Dynamic VM Consolidation (8/13)

- Host Underloading Detection
  - Simple approach
    - System finds the host with the minimum utilization, and try to migrate VMs to other hosts keeping them not overloaded.
    - If this can be accomplished:
      - Migrate all the VMs, and switch to sleep mode.
      - The iteration continues.
    - Else:
      - Host keeps alive. And the iteration breaks.

Adaptive Heuristics for Dynamic VM Consolidation (9/13)

- VM Selection
  - Minimum Migration Time (MMT)
  - Random Choice (RC)
  - Maximum Correlation (MC)

Adaptive Heuristics for Dynamic VM Consolidation (10/13)

- Minimum Migration Time (MMT)
  - Migrates a VM that requires the minimum time to complete compared to the other VMs allocated to the host.

$$T_{j,host} = \frac{Memory_j}{Bandwidth_{host}}$$

$$v \in V_j | \forall a \in V_j, T_{v,j} \leq T_{a,j}$$

Adaptive Heuristics for Dynamic VM Consolidation (11/13)

- Random Choice (RC)
  - Just choose a random VM to migrate.

$$X \stackrel{\text{def}}{=} U(0, |V_j|)$$

Adaptive Heuristics for Dynamic VM Consolidation (12/13)

- Maximum Correlation (MC)
  - Based on the idea proposed by Verma et al.
  - We select those VMs to be migrated that have the highest correlation of the CPU utilization with other VMs.

$$X = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1,1} & \cdots & x_{n-1,n-1} \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$$\hat{y} = Xb \quad b = (X^T X)^{-1} X^T y$$

$$R^2_{Y,X_1,\cdots,X_{n-1}} = \frac{\sum_{i=1}^{n}(y_i - m_Y)^2(\hat{y}_i - m_{\hat{Y}})^2}{\sum_{i=1}^{n}(y_i - m_Y)^2 \sum_{i=1}^{n}(\hat{y}_i - m_{\hat{Y}})^2}$$

$$v \in V_j | \forall a \in V_j, R^2_{X_v,X_1,\cdots,X_{v-1},X_{v+1},\cdots,X_n} \geq R^2_{X_v,X_1,\cdots,X_{a-1},X_{a+1},\cdots,X_n}$$

Adaptive Heuristics for Dynamic VM Consolidation (13/13)

- VM Placement

Algorithm 2: Power Aware Best Fit Decreasing (PABFD)

```
1  Input: hostList, vmList   Output: allocation of VMs
2  vmList.sortDecreasingUtilization()
3  foreach vm in vmList do
4      minPower ← MAX
5      allocatedHost ← NULL
6      foreach host in hostList do
7          if host has enough resources for vm then
8              power ← estimatePower(host, vm)
9              if power < minPower then
10                 allocatedHost ← host
11                 minPower ← power
12     if allocatedHost ≠ NULL then
13         allocation.add(vm, allocatedHost)
14 return allocation
```

Performance Evaluation (1/12)

- CloudSim 2.0
  - 1 data center
  - 800 heterogeneous physical nodes
    - 400 HP ProLiant ML110 G4
      - 1860 MIPS x 2
      - 1GB/s network bandwidth
    - 400 HP ProLiant ML110 G5
      - 2660 MIPS x 2
      - 1GB/s network bandwidth

Performance Evaluation (2/12)

- 500 heterogeneous VM requests
  - Amazon EC2 Instances **(*except all VMs are single-core)** (http://aws.amazon.com/ec2/#instance)
  - High-CPU Medium Instance
    - 2500 MIPS, 0.85 GB
  - Extra Large Instance
    - 2000 MIPS, 3.75 GB
  - Small Instance
    - 1000 MIPS, 1.7 GB
  - Micro Instance
    - 500 MIPS, 613 MB

Performance Evaluation (3/12)

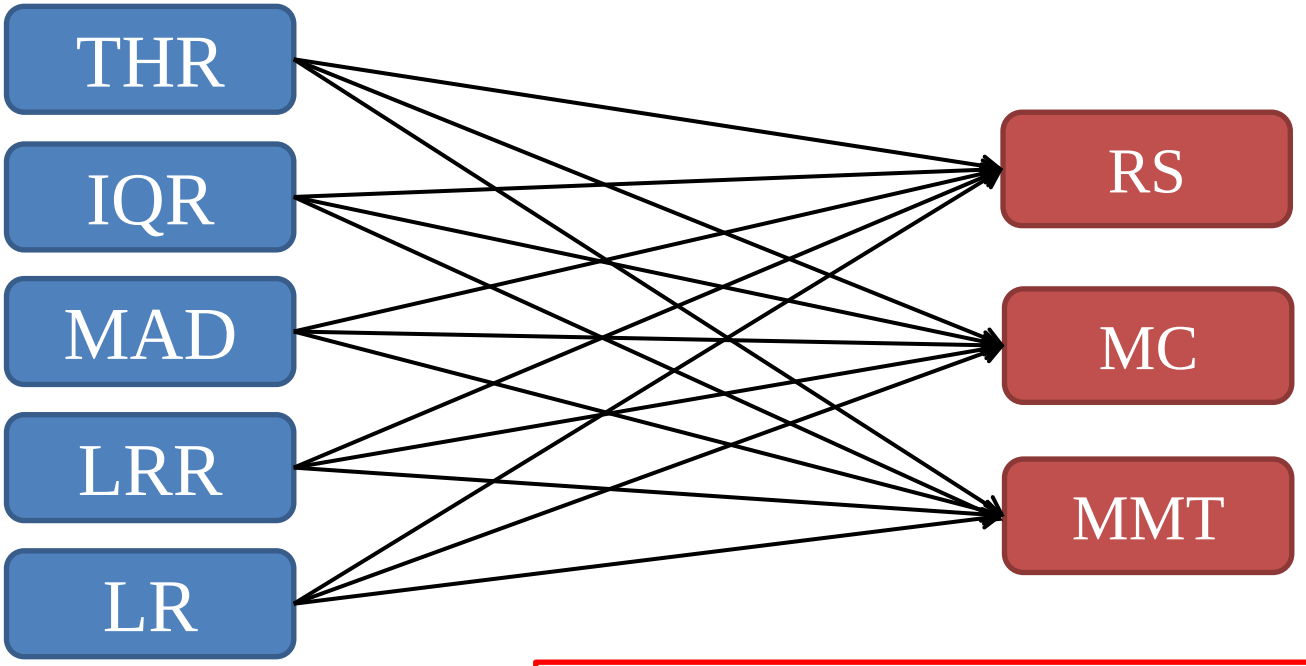
- Workload Data
  - CoMon Project from PlanetLab
  - Each VM is randomly assigned a workload data

Table II. Workload data characteristics (CPU utilization)

Date	Number of VMs	Mean	St. dev.	Quartile 1	Median	Quartile 3
03/03/2011	1052	12.31%	17.09%	2%	6%	15%
06/03/2011	898	11.44%	16.83%	2%	5%	13%
09/03/2011	1061	10.70%	15.57%	2%	4%	13%
22/03/2011	1516	9.26%	12.78%	2%	5%	12%
25/03/2011	1078	10.56%	14.14%	2%	6%	14%
03/04/2011	1463	12.39%	16.55%	2%	6%	17%
09/04/2011	1358	11.12%	15.09%	2%	6%	15%
11/04/2011	1233	11.56%	15.07%	2%	6%	16%
12/04/2011	1054	11.54%	15.15%	2%	6%	16%
20/04/2011	1033	10.43%	15.21%	2%	4%	12%

Performance Evaluation (4/12)

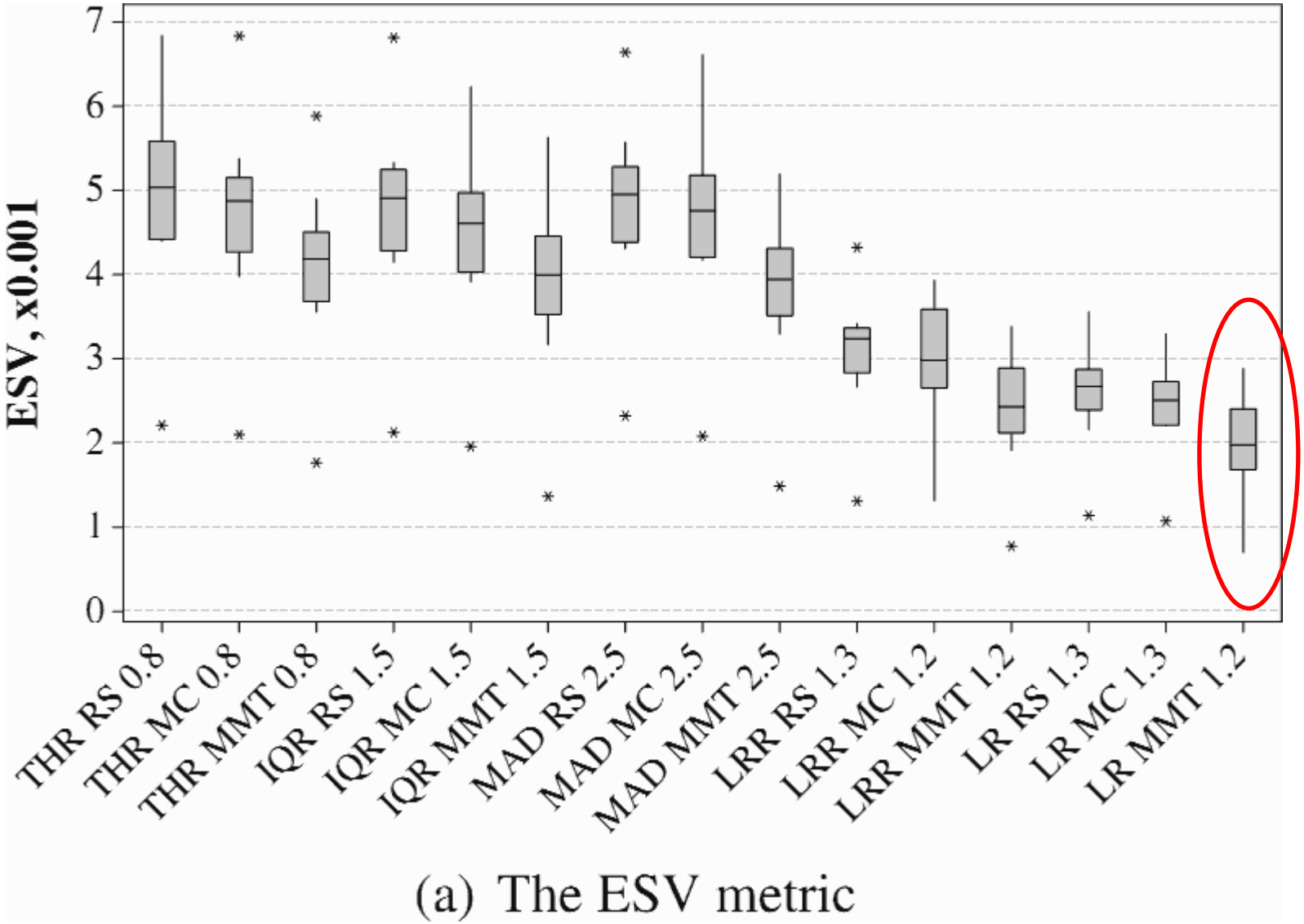
5 host overloading detection algorithms



3 VM selection algorithms

- THR RS 0.8
- THR MC 0.8
- THR MMT 0.8
- IQR RS 1.5
- IQR MC 1.5
- IQR MMT 1.5
- MAD RS 2.5
- MAD MC 2.5
- MAD MMT 2.5
- LRR RS 1.3
- LRR MC 1.2
- LRR MMT 1.2
- LR RS 1.3
- LR MC 1.3
- LR MMT 1.2

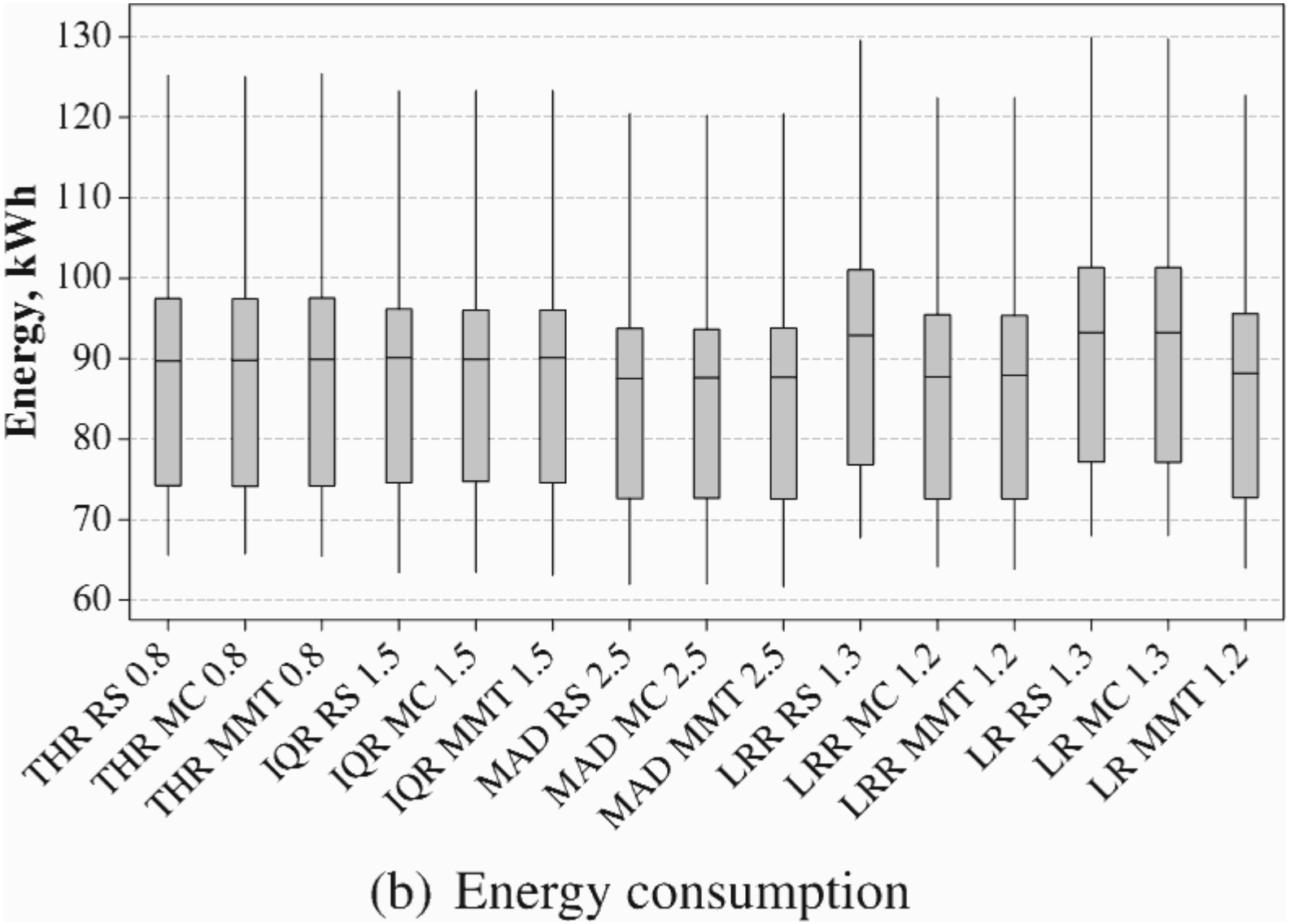
Performance Evaluation (5/12)



(a) The ESV metric

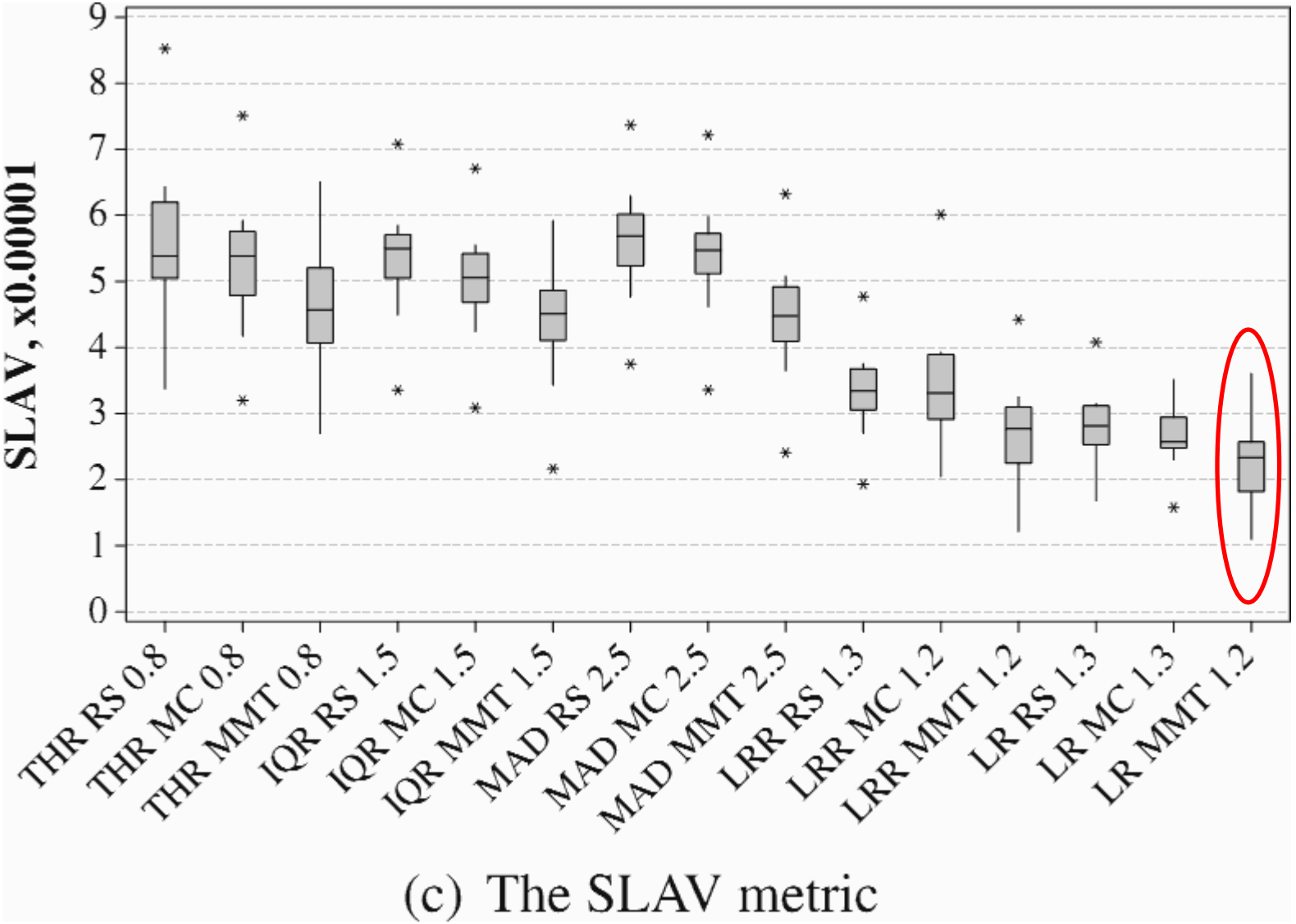
$$ESV = E \cdot SLAV$$

Performance Evaluation (6/12)



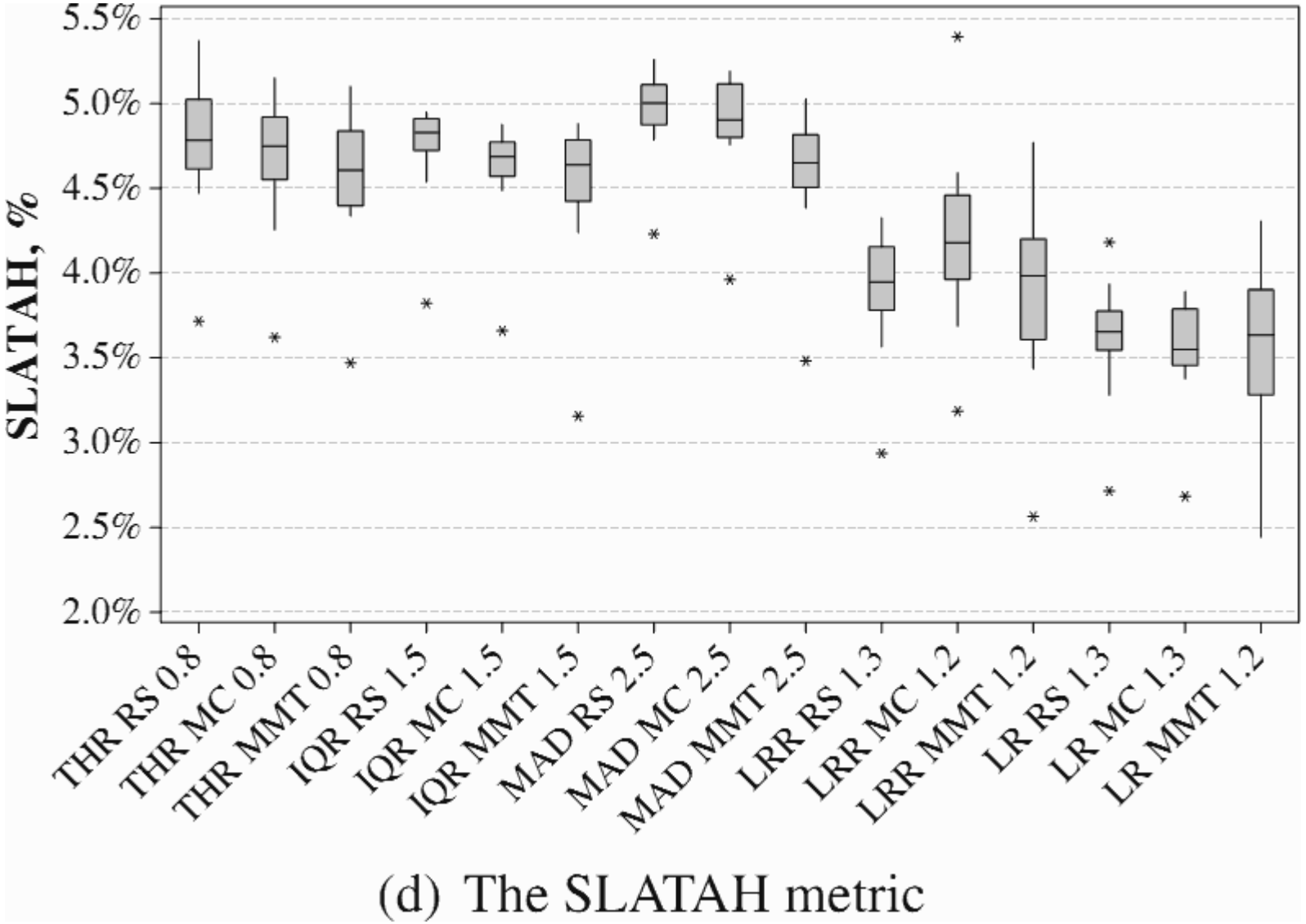
(b) Energy consumption

Performance Evaluation (7/12)



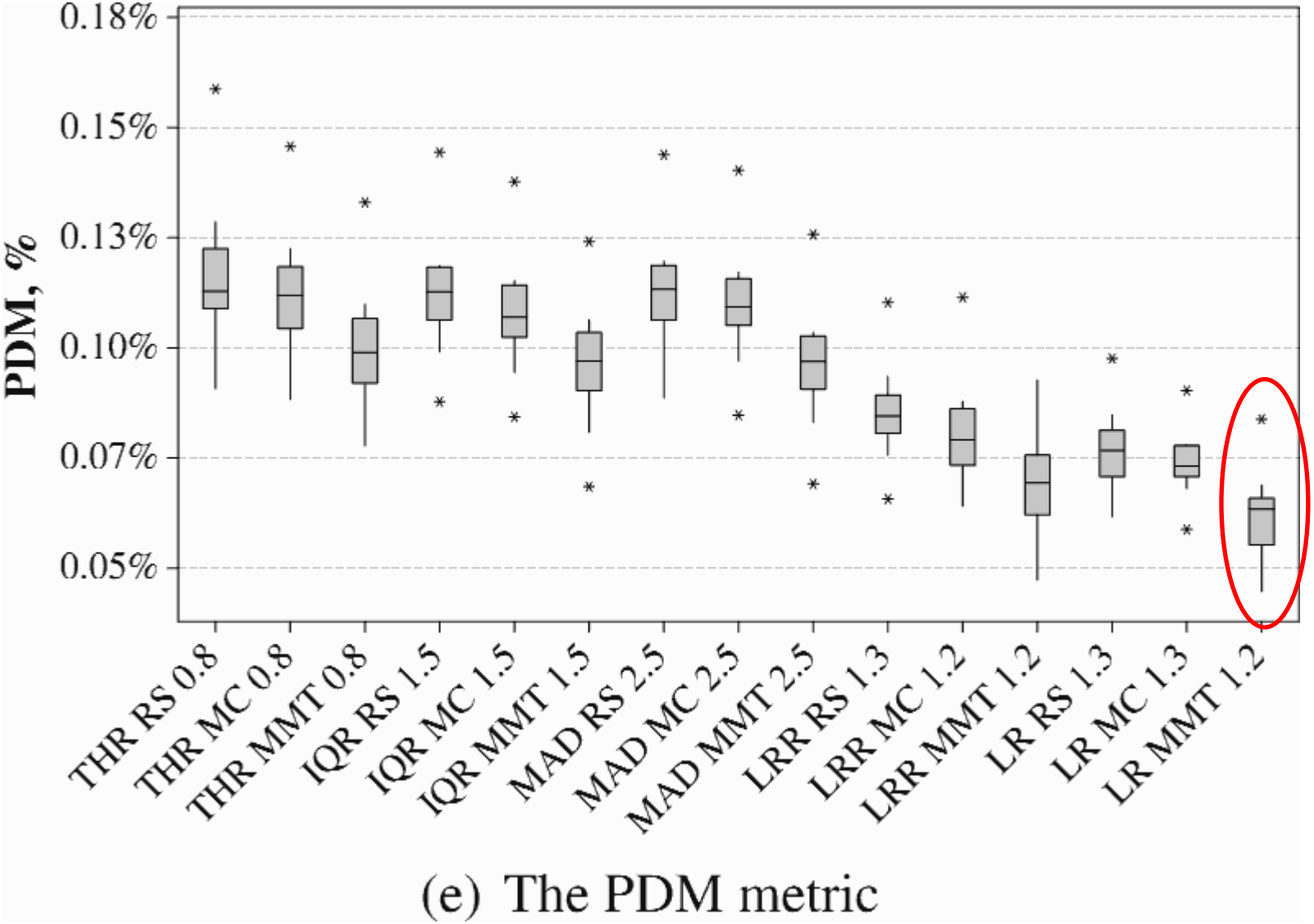
(c) The SLAV metric

Performance Evaluation (8/12)



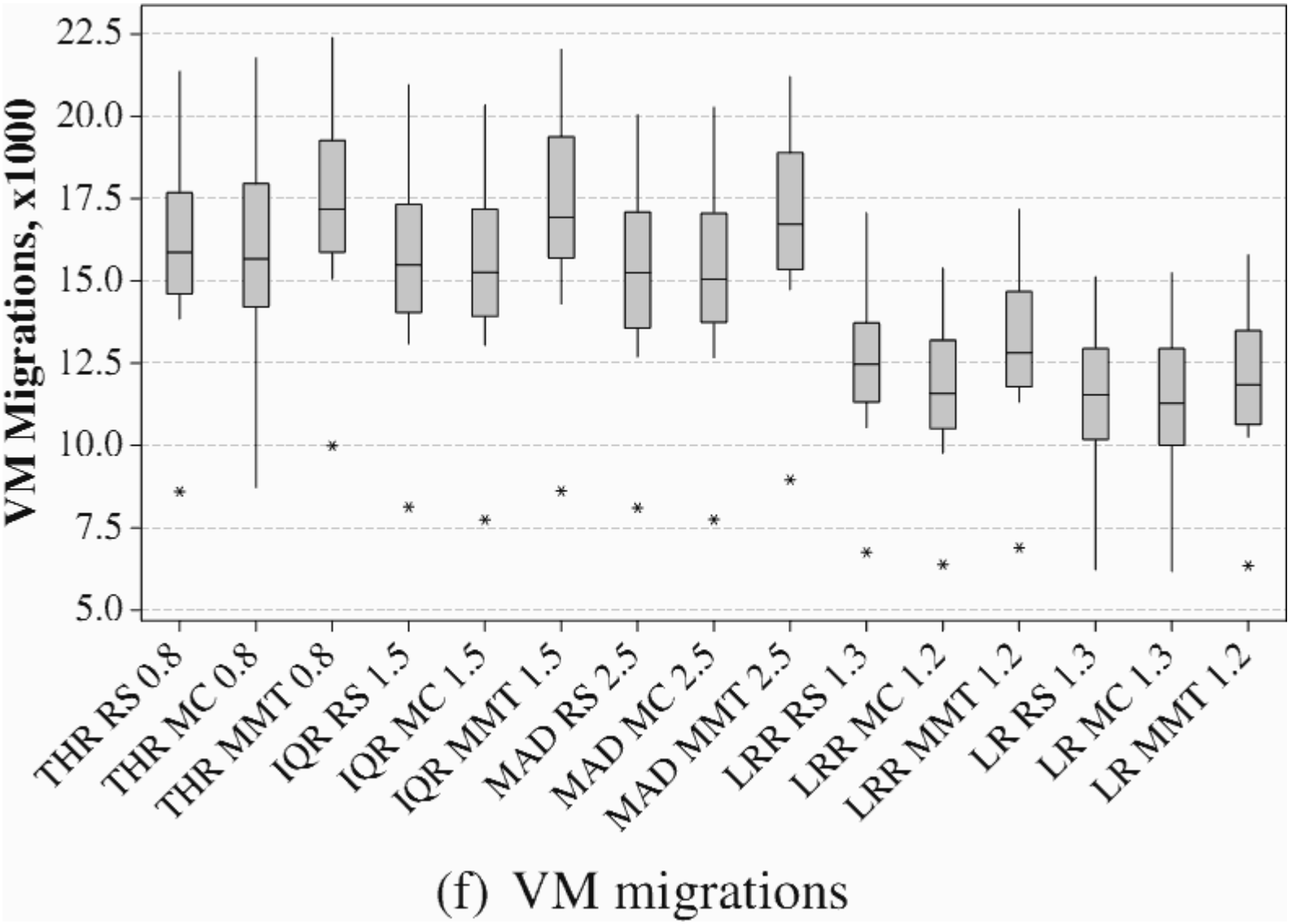
(d) The SLATAH metric

Performance Evaluation (9/12)



(e) The PDM metric

Performance Evaluation (10/12)



(f) VM migrations

Performance Evaluation (11/12)

Table V. Simulation results of the best algorithm combinations and benchmark algorithms (median values)

Policy	**ESV** ($\times 10^{-3}$)	**Energy** (kWh)	**SLAV** ($\times 10^{-5}$)	**SLATAH**	**PDM**	**VM migr.** ($\times 10^{3}$)
NPA	0	2419.2	0	0%	0%	0
DVFS	0	613.6	0	0%	0%	0
THR-MMT-1.0	20.12	75.36	25.78	24.97%	0.10%	13.64
THR-MMT-0.8	4.19	89.92	4.57	4.61%	0.10%	17.18
IQR-MMT-1.5	4.00	90.13	4.51	4.64%	0.10%	16.93
MAD-MMT-2.5	3.94	87.67	4.48	4.65%	0.10%	16.72
LRR-MMT-1.2	2.43	87.93	2.77	3.98%	0.07%	12.82
LR-MMT-1.2	1.98	88.17	2.33	3.63%	0.06%	11.85

Performance Evaluation (12/12)

- Observations:
  1. Dynamic VM consolidation algorithms are better than static allocation DVFS/NPA policies.
  2. MMT is better than RS and MC, meaning that the minimization of the VM migration time is more important.
  3. Local regression has better predictions of host overloading than adaptive-threshold based algorithms.

Conclusion and Future Directions (1/2)

- To maximize the ROI, Cloud providers have to apply energy-efficient strategies such as dynamic VM consolidation and switching idle servers to power-saving modes.

- However, such consolidation is not trivial, as it can result in violation of SLA.

- We have proposed novel adaptive heuristics based on an analysis of historical data on the resource usage for energy and performance efficient dynamic consolidation of VMs.

Conclusion and Future Directions (2/2)

- The evaluation shows the proposed local regression (LR) based algorithm combined with the MMT VM selection policy outperforms other algorithms.

- We plan to implement it in a real-world Cloud platform, such as OpenStack. Another direction for research is the investigation of more complex workload model.